Integer-Only Cross Field Computation

NAHUM FARCHI and MIRELA BEN-CHEN, Technion Institute of Technology

We propose a new iterative algorithm for computing smooth cross fields on triangle meshes that is simple, easily parallelizable on the GPU, and finds solutions with lower energy and fewer cone singularities than state-of-the-art methods. Our approach is based on a formal equivalence, which we prove, between two formulations of the optimization problem. This equivalence allows us to eliminate the real variables and design an efficient grid search algorithm for the cone singularities. We leverage a recent graph-theoretical approximation of the *resistance distance matrix* of the triangle mesh to speed up the computation and enable a trade-off between the computation time and the smoothness of the output.

CCS Concepts: • **Computing methodologies** → **Mesh models**;

Additional Key Words and Phrases: cross fields, singularities, quad remeshing, digital geometry processing, resistance distance

ACM Reference Format:
Nahum Farchi and Mirela Ben-Chen, Technion Institute of Technology. 2018. Integer-Only Cross Field Computation. *ACM Trans. Graph.* 37, 4, Article 91 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201375

1 INTRODUCTION

Directional fields, and especially *cross fields*, are important objects in geometry processing. They are used in many applications, from quadrangular remeshing to non-photorealistic rendering [Vaxman et al. 2016]. Computing smooth cross fields on triangle meshes is challenging, as the problem formulation inherently depends on *integer* variables to encode the invariance of the crosses to rotations by integer multiples of $\pi/2$.

A popular approach, suggested by Bommes et al. [2009] (MIQ), formulates a mixed-integer optimization problem and solves it greedily to compute the cross field. While highly efficient and effective, the greedy solution can lead to sub-optimal results, as in Fig. 1 (top). Alternatively, Crane et al. [2010] (TCODS) [Crane et al. 2010] posed the problem in terms of *angle defects* due to parallel transport on closed cycles, leading to a sparse linear least squares problem that is solved efficiently when the defects are known.

We show that if the angle defects are *unknown*, and there are no directional constraints, these two optimization problems are *equivalent*. Furthermore, by eliminating the real variables, we remain with an *integer only* optimization problem. We use this insight to design a new iterative algorithm for minimizing the energy that is simple, easily parallelizable on the GPU, and finds solutions with lower energy and fewer singularities than MIQ, e.g. Fig. 1 (bottom). Finally, we show the connection of the minimized energy to the *resistance distance matrix* of the triangle mesh, and leverage a recent graph theoretical approximation to speed up the computation and allow us to trade-off the computation time and the quality of the resulting cross field.

Author's address: Nahum Farchi and Mirela Ben-Chen, Technion Institute of Technology.

1.1 Related Work

Cross field computation, and directional field computation in general, has seen a surge of research in recent years. A recent review [Vaxman et al. 2016] covers the latest developments, and we therefore focus our literature review on methods closest to our approach.

Angle based representation. A popular formulation of the cross field computation problem is to represent every cross as an *angle* with respect to a fixed local orthogonal frame. Since crosses are invariant to rotations by integer multiples of $\pi/2$, such a representation has an inherent phase ambiguity. Therefore, finding a *smooth* assignment of crosses inevitably requires taking into account these unknown integer phases, leading to optimization problems with integer variables. Bommes et al. [2009] suggested one of the first efficient methods to tackle these optimization problems in the context of cross field generation, by greedily rounding to an integer one variable per iteration and resolving the system. Our approach optimizes the same energy greedily, albeit using a different algorithm that guarantees that there exists no modification of a single $\pm\frac{\pi}{2}$ singularity's position that reduces the energy. This leads to lower energy values and better singularity placement. A different angle

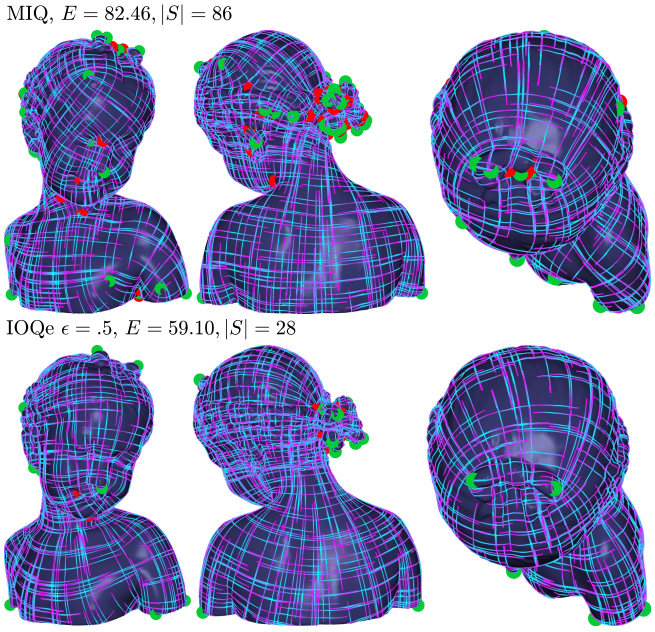

Fig. 1. Our iterative optimization (bottom) finds a solution with lower energy, and fewer singularities than MIQ (top).

based approach, suggested by Crane et al. [2010], encodes the *angle difference* per edge instead of an angle per face. This representation leads to a minimum norm linear least squares optimization problem with constraints, where the integer variables now arise as the constrained values. Furthermore, Crane et al. [2013, Sec. 8.4.1] have shown that it is possible to solve this optimization problem by solving a single Poisson problem when the integer variables are known. We use this formulation with *unknown* angle defect values as the basis for our algorithm.

Cartesian representation. An alternative to the angle based representation is to represent the direction as two coordinate values with respect to a local frame [Ray et al. 2006] or, equivalently, as a complex number [Knöppel et al. 2013]. The explicit encoding of the integer phase is not required in this representation, albeit, depending on the choice of smoothness energy, a non-convex pointwise unit-length constraint might be required. Without the unit-length constraint, this formulation leads to an unconstrained linear least squares problem that can be efficiently and globally solved [Knöppel et al. 2013]. Our main interest is in the angle-based energy, as it has various advantages in applications; see [Vaxman et al. 2016]. We show that for this energy our algorithm achieves lower energy values, with a smaller number of singularities, compared to competing approaches.

Scalable cross field computation. Recently, new methods have been proposed [Jakob et al. 2015] for efficient cross field computation that are applicable to meshes with millions of triangles. Such approaches often work locally, leading to a very efficient solution at the price of cross field quality in terms of the number of singularities and field smoothness. Our approach is at the other end of the spectrum, namely, we invest more computational time and generate a higher quality cross field. We further allow a trade-off between computational time and cross field quality using a single parameter. Finally, our time/quality trade-off is implemented using a simple algebraic approach with random projections, and does not require constructing multi-resolution hierarchies of the input shape.

Parameterization with cone singularities. Cross field computation is closely related to mesh parameterization. Specifically, one of the main applications of cross fields is quadrangular remeshing, where the parameterization gradients are aligned to the cross directions. Then, the singularities of the cross field become the non-regular vertices of the quad mesh. Hence, it is in general beneficial to generate smooth cross fields with a small number of singularities. As an alternative to generating a cross field and using it for creating a parameterization, it is possible to compute a parameterization with *cone singularities* given a *holonomy signature*. Such conformal parameterizations were suggested [Ben-Chen et al. 2008; Springborn et al. 2008], as well as variants that use other energies [Myles and Zorin 2012, 2013], guarantee bijectivity [Bright et al. 2017] or generate a seamless similarity map that can be used for constructing C^2 surfaces [Campen and Zorin 2017b]. While our approach generates cross fields, it is based on finding a holonomy signature, and thus can be used to generate inputs for cone parameterization methods such as [Bright et al. 2017; Campen and Zorin 2017b].

Connectivity Editing. Peng et al. [2011] have proposed a set of edit operations on a convex region of the quadrangular mesh to improve the placement of irregular vertices (i.e,. vertices with valence different than four). For example, they show that the global placement of a single irregular vertex is in some sense rigid, whereas singularity pairs in close proximity can be locally moved to improve the structure of the quadrangular mesh. In contrast, our approach guarantees that no movement of a single singularity, or the global cancellation of a $\pm\pi/2$ singularity pair can improve the energy. It would be interesting to explore their other suggested edit operations to locally improve the quadrangular mesh structure after generating the global structure using our method.

1.2 Contributions

We show the equivalence between computing smooth cross fields and finding optimal holonomy signatures in the absence of directional constraints, and leverage it to design a novel algorithm that optimizes the angle-based cross field smoothness energy. Our approach has the following advantages:

- The algorithm is simple, easily parallelizable and finds cross fields with lower energy values than existing approaches.
- The output cross fields are such that there is no relocation of a single $\pm\frac{\pi}{2}$ singularity that will reduce the energy. This leads to cross fields with fewer singularities, and singularities that are better placed, compared to existing methods.
- The formulation is based on the resistance distance matrix, which has a well-known random approximation with theoretical guarantees. We use this approximation to trade-off between cross field smoothness and computation time.

2 BACKGROUND: ANGLE-BASED CROSS FIELD COMPUTATION

Notation. Let $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ be a 2-manifold closed orientable triangle mesh, where $\mathcal{V}$ are the vertices, $\mathcal{E}$ are the edges and $\mathcal{F}$ are the faces. We denote $n = |\mathcal{V}|, l = |\mathcal{E}|, m = |\mathcal{F}|$, the genus of $\mathcal{M}$ by g, and its Euler characteristic by $\chi = 2 - 2g$. We further denote the *dual* mesh by $\mathcal{M}^* = (\mathcal{V}^*, \mathcal{E}^*, \mathcal{F}^*) = (\mathcal{F}, \mathcal{E}^*, \mathcal{V})$. Following existing work, see e.g. [Vaxman et al. 2016, Sec. 5.1], we represent crosses using angles. Thus, we use $\theta \in \mathbb{R}^m$ to denote angles on the faces, which are measured relative to a local frame of reference, i.e., a pair of orthogonal unit vectors tangent to the face. We further assume that each edge in $\mathcal{E}$ has a known, arbitrary orientation that also induces an orientation on the corresponding dual edge. We denote by $r \in \mathbb{R}^l$ the *oriented* angle difference between the reference frames on adjacent faces. We slightly abuse notation by addressing elements of r both as r_e and as r_{ij} where $e = (i, j) \in \mathcal{E}^*, i, j \in \mathcal{F}$. Finally, $d_0 \in \mathbb{Z}^{l \times n}$ and $d_1 \in \mathbb{Z}^{m \times l}$ denote the edge-vertex and face-edge adjacency matrices, respectively, also known as the *discrete exterior derivatives* on 0- and 1-forms [Crane et al. 2013].

A natural way to define the smoothness of an angle-based cross field is to consider the change in the angle between adjacent faces. Two methods that were suggested in the literature, MIQ [Bommes et al. 2009] and TC [Crane et al. 2010], approach this problem using different formulations. In the following, we first present the two optimization problems as they were originally suggested. Then, in

Section 3 we generalize TC, and show that the new formulation is equivalent to MIQ, yet simpler to optimize. We provide only a brief overview of the methods, and refer to specific sections of the survey [Vaxman et al. 2016] and course [Crane et al. 2013] for basic concepts.

2.1 MIQ: Mixed Integer Quadrangulation

Bommes et al. [2009] represented a cross field by an angle per face, $\theta \in \mathbb{R}^m$, relative to a fixed local orthogonal frame. To account for the symmetry of the crosses with respect to rotations by $\pi/2$, additional *period jumps*, $p \in \mathbb{Z}^l$, were introduced. The MIQ objective function is given by:

$$E_M(\theta, p) = \sum_{(i,j)\in\mathcal{E}^*} \left(\theta_i + r_{ij} + \tfrac{\pi}{2} p_{ij} - \theta_j\right)^2. \tag{1}$$

We will assume a single directional constraint is given at a face $c \in \mathcal{F}$, with $\theta_c = \theta_0$. This objective function has multiple minimizers, which can be obtained by modifying θ and p simultaneously. Therefore, to reduce the search space, Bommes et al. [2009] used a spanning tree $\mathcal{T} \subset \mathcal{E}^*$ of the dual mesh $\mathcal{M}^*$, rooted at the constrained face c, and defined the optimization problem:

$$\begin{aligned} \underset{\theta\in\mathbb{R}^m,\, p\in\mathbb{Z}^l}{\text{minimize}} \quad & E_M(\theta, p) \\ \text{subject to} \quad & p_e = 0, \quad \forall e \in \mathcal{T}, \\ & \theta_c = \theta_0. \end{aligned} \tag{2}$$

Effectively, the constraints can be easily eliminated, leading to an unconstrained mixed-integer problem in $m-1$ real-valued variables and $l-m+1 = n+2g-1$ integer-valued variables.

2.2 TC: Trivial Connections

Alternatively, instead of solving for the angles on the faces, Crane et al. [2010] suggested to solve for the *adjustment angles*, or *connection* on the edges. As these define the *change* in the angle when passing on a dual edge [Vaxman et al. 2016, Sec.4.3], explicitly encoding the period jumps is not required. Hence, the real-valued variables $x \in \mathbb{R}^l$ encode the change in angle, and the objective function is given by:

$$E_T(x) = \|x\|_2^2. \tag{3}$$

The angles θ are obtained by integrating x along a dual tree $\mathcal{T}$ rooted at the constrained face c, such that $\theta_j = \theta_i + r_{ij} + x_{ij}$ for $(i,j) \in \mathcal{T} \subset \mathcal{E}^*$.

While this objective function does not depend on integer variables, not every $x \in \mathbb{R}^l$ is valid, as different integration paths should yield the same angle up to rotation by $\pi/2$. Thus, additional constraints are required, leading to the optimization problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^l}{\text{minimize}} \quad & E_T(x) \\ \text{subject to} \quad & \Gamma x = \tfrac{\pi}{2} s - s_g(\Gamma). \end{aligned} \tag{4}$$

Here, $\Gamma \in \mathbb{Z}^{n+2g\times l}$ is a matrix whose rows form a spanning set of the dual cycles of $\mathcal{M}$. Specifically, $\Gamma^T = \left[d_0, H\right]$, where $d_0 \in \mathbb{Z}^{l\times n}$ is the oriented edge-vertex incidence matrix, whose columns form a spanning set of the *contractible* dual cycles, and $H \in \mathbb{Z}^{l\times 2g}$ is a matrix whose columns form a basis for the non-contractible dual cycles (see [Crane et al. 2013, Sec. 8.2.2] for the construction of H). Further, $s_g(\Gamma) \in \mathbb{R}^{n+2g}$ contains the *angle defects* [Vaxman et al. 2016, Sec.6.2] around the basis cycles of Γ. The angle defects for the contractible cycles are given by the discrete Gaussian curvature of the vertices, and thus sum to $2\pi\chi$ by the discrete Gauss-Bonnet formula [Meyer et al. 2003].

Finally, $s \in \mathbb{Z}^{n+2g}$ is a user prescribed integer *holonomy signature* that defines the number of integer rotations by $\pi/2$ when parallel transporting a vector along the dual cycles in Γ. Since every column of d_0^T sums to 0, for the constraints to be feasible it is assumed that $\sum_{i=1}^n s_i = 4\chi$. Crane et al. [2010] showed that under this assumption the optimization problem (4) always has a solution, and a singularity of the cross field will arise at a vertex $v_i \in \mathcal{V}$, $i \in \{1, .., n\}$ if and only if $s_i \neq 0$, i.e., the prescribed holonomy signature of the corresponding contractible dual cycle is non-zero.

3 INTEGER-ONLY CROSS FIELD COMPUTATION

3.1 TCO: Trivial Connections with Optimal Holonomies

A natural generalization of the TC approach is to add the integer holonomies as optimization variables instead of having the user prescribe them. This generalization leads to the optimization problem:

$$\begin{aligned} \underset{x\in\mathbb{R}^l,\, \alpha\in\mathbb{Z}^n,\, \beta\in\mathbb{Z}^{2g}}{\text{minimize}} \quad & E_T(x) \\ \text{subject to} \quad & \begin{bmatrix} d_0^T \\ H^T \end{bmatrix} x - \tfrac{\pi}{2} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = - \begin{bmatrix} \alpha_g \\ \beta_g(H) \end{bmatrix}, \\ & \textstyle\sum_{i=1}^n \alpha_i = 4\chi. \end{aligned} \tag{5}$$

Here, for notational convenience, we separate the holonomy signature as $s = \left[\alpha, \beta\right]$, where α will denote the cone singularities vector and β the angle defects on non-contractible cycles. Similarly, α_g is the discrete Gaussian curvature, and $\beta_g(H)$ the geometric angle defects of the dual cycles in H, where both are computable from the geometry of the input mesh (see Figure 2).

A main result of this paper is that the optimization problems in Equations (2) and (5) are equivalent. Formally, we have:

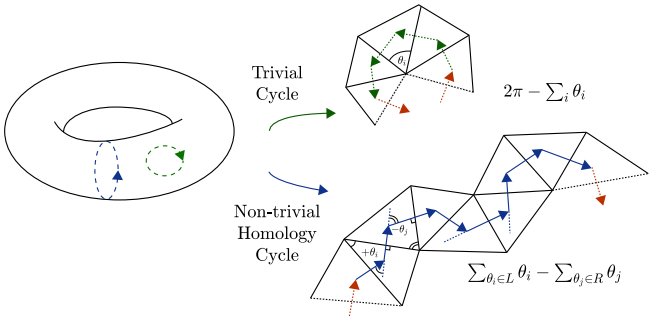

Fig. 2. Computing the angle defects for (green) trivial and (blue) non-trivial cycles. For trivial cycles, we take 2π minus the sum of angles, which is the usual discrete Gaussian curvature. For non-trivial cycles, we take the sum of exterior angles along the blue curve–adding θ whenever the curve turns left, and subtracting θ whenever the curve turns right.

Theorem 3.1.

(i) Let (θ, p) be a feasible solution of (2). Then, for any integral basis of non-contractible dual cycles H, there exists a feasible solution (x, α, β) of (5) such that $E_T(x) = E_M(\theta, p)$.

(ii) Let (x, α, β) be a feasible solution of (5). Then, for any dual spanning tree $\mathcal{T}$, there exists a feasible solution (θ, p) of (2) such that $E_M(\theta, p) = E_T(x)$.

(iii) Let (x, α, β) and (θ, p) be corresponding solutions as in (i,ii), and let θ_T be the integrated values of x. Then $\theta_T = \theta \mod \pi/2$.

A proof is given in the Appendix. The main building block of the proof is to relate the variables of the two optimization problems using a linear system of equations. We show that this system always has a unique solution that is integer-valued for p and (α, β). The integer solutions are guaranteed by a result from the theory of cycle bases on graphs [Liebchen and Rizzi 2007], stating that a square submatrix of an integral cycle basis matrix, obtained by removing columns corresponding to edges of a spanning tree, is unimodular.

Corollary 3.2. *The optimization problems MIQ and TCO are invariant to the choice of dual spanning tree $\mathcal{T}$ and basis for non-contractible cycles H, respectively. Specifically:*

(i) Given a feasible solution (θ, p) to MIQ with some spanning tree $\mathcal{T}$, then for any spanning tree $\tilde{\mathcal{T}}$, there exists a feasible solution $(\tilde{\theta}, \tilde{p})$ such that $E_M(\theta, p) = E_M(\tilde{\theta}, \tilde{p})$ and $\tilde{\theta} = \theta \mod \frac{\pi}{2}$.

(ii) Given a feasible solution (x, α, β) to TCO with some choice of basis H, then for any basis $\tilde{H}$, there exists a feasible solution $(\tilde{x}, \tilde{\alpha}, \tilde{\beta})$, such that $E_T(x) = E_T(\tilde{x})$ and $\tilde{\theta}_T = \theta_T \mod \frac{\pi}{2}$.

This is a straightforward result of Theorem 3.1: given a solution (θ, p) to MIQ with some spanning tree $\mathcal{T}$, we use part (i) of the theorem to construct a solution (x, α, β) to TCO, and then use part (ii) with a *different* spanning tree $\tilde{\mathcal{T}}$ to construct another MIQ solution $(\tilde{\theta}, \tilde{p})$. The theorem guarantees that $E_M(\theta, p) = E_M(\tilde{\theta}, \tilde{p})$ and also that $\tilde{\theta} = \theta \mod \frac{\pi}{2}$. A similar argument shows that TCO is invariant to the choice of H. Note that p and β might change, though this is inconsequential to the resulting cross fields, which are given by θ and θ_T respectively.

As the optimization problems are equivalent, we can devise an algorithm for optimizing Equation (5) instead of Equation (2). There are a few advantages to changing the parameterization of the problem to the variables (x, α, β). First, we can use the discrete Hodge decomposition [Tong et al. 2003] to eliminate the real variables x and remain with an integer-only problem. Second, the integer variables α have a geometric meaning, as the cone singularities of the computed cross field, and thus we can devise an efficient iterative method for optimizing them. Finally, the separation of α and β allows us to relax β while optimizing α, simplifying the algorithm for high genus meshes.

3.2 IOQ

The problem in Equation (5) has some interesting properties, as was noted in [Crane et al. 2013, Sec. 8.4.1]. First, recall a fundamental property of the adjacency matrices d_0, d_1, namely that $d_1 d_0 = 0$. Hence, any vector $x \in \mathbb{R}^l$ can be uniquely decomposed as $x = d_0 a + Bb + d_1^T c$, where $B \in \mathbb{R}^{l \times 2g}$ is a matrix whose columns form a basis for the linear space $\ker(d_1) \setminus \mathrm{im}(d_0)$, and $a \in \mathbb{R}^n, b \in \mathbb{R}^{2g}, c \in \mathbb{R}^m$. B is computed from H by $B = H - d_0 (d_0^T d_0)^\dagger d_0^T H$ and is orthogonal to d_0 and d_1 (see [Crane et al. 2013, Sec. 8.2.2]). Here $\dagger$ indicates the Moore-Penrose pseudo-inverse. This decomposition is also known as the Hodge decomposition of discrete differential forms [Tong et al. 2003] (we discuss the metric in Section 6.1). Thus, the constraint in Equation (5) can be written as:

$$\begin{bmatrix} d_0^T \\ H^T \end{bmatrix} \begin{bmatrix} d_0 & B & d_1^T \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \tfrac{\pi}{2} s - s_g(\Gamma), \tag{6}$$

yielding the constraint matrix

$$\begin{bmatrix} d_0^T d_0 & d_0^T B & d_0^T d_1^T \\ H^T d_0 & H^T B & H^T d_1^T \end{bmatrix} = \begin{bmatrix} d_0^T d_0 & 0 & 0 \\ H^T d_0 & H^T B & 0 \end{bmatrix}. \tag{7}$$

Here we used the fact that the matrices d_0, B are orthogonal, and the fact that the edge values of the non-contractible dual cycles in H sum to 0 on all triangles; thus $H^T d_1^T = 0$. Consequently, c is not constrained by Equation (7). Due to the orthogonality of the decomposition, we have $E_T(x) = \|d_0 a\|_2^2 + \|Bb\|_2^2 + \|d_1^T c\|_2^2$, and thus the optimal solution will always have $c = 0$. Finally, the constraint on a does not depend on β, and is given by:

$$a(\alpha) = L^\dagger (\tfrac{\pi}{2} \alpha - \alpha_g), \tag{8}$$

where $L = d_0^T d_0$ is the graph Laplacian. Note that $a(\alpha)$ is defined only up to an additive constant, since L has co-rank 1. Similarly, the constraint on b is:

$$b(\alpha, \beta) = (H^T B)^{-1} \left(\tfrac{\pi}{2} \beta - \beta_g(\Gamma) - H^T d_0 a(\alpha) \right), \tag{9}$$

where $H^T B$ is non-singular since both H and B are full rank.

Combining these properties allows us to eliminate the real-valued variables a, b and remain only with the integer-valued variables α, β. Hence, the part of the objective function that depends on the cone singularities α is:

$$E_I(\alpha) = \|d_0 a(\alpha)\|_2^2 = (\tfrac{\pi}{2} \alpha - \alpha_g)^T L^\dagger (\tfrac{\pi}{2} \alpha - \alpha_g), \tag{10}$$

where we used the fact that $L^\dagger$ is symmetric, and $L^\dagger L L^\dagger = L^\dagger$. Finally, the optimization problem is:

$$\begin{aligned} \underset{\alpha \in \mathbb{Z}^n, \beta \in \mathbb{Z}^{2g}}{\text{minimize}} \quad & E_I(\alpha, \beta) = E_I(\alpha) + \|Bb(\alpha, \beta)\|_2^2 \\ \text{subject to} \quad & \textstyle\sum_{i=1}^n \alpha_i = 4\chi. \end{aligned} \tag{11}$$

The optimization problems TCO and IOQ are equivalent. Formally, we have:

Theorem 3.3. *(x, α, β) is an optimal solution to Equation (5) if and only if $x = d_0 a(\alpha) + Bb(\alpha, \beta)$ and (α, β) is an optimal solution to Equation (11).*

The proof is a straightforward result of Equations (6)-(9) and is provided in the Appendix for completeness. Similar results, albeit not in the context of optimizing the holonomy signature, appear in [Crane et al. 2013, Sec. 8.4.1] and [Campen and Zorin 2017a].

Corollary 3.4. *For a closed, oriented triangle mesh, with a single directional constraint, the optimal cross field for MIQ, i.e., the optimal $\theta \mod \frac{\pi}{2}$ in Equation (2), is fully determined by the holonomy signature $s = [\alpha, \beta]$.*

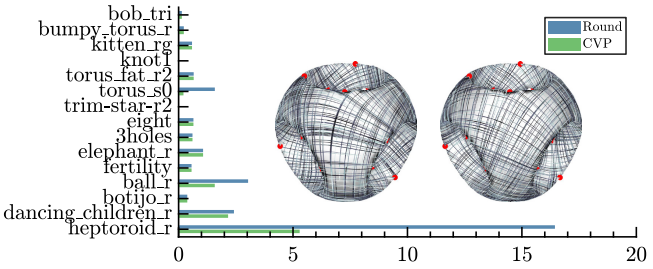

Fig. 3. Comparison between direct rounding and an exact CVP solver [Agrell et al. 2002] for some low resolution meshes sorted by genus. Note that in most cases direct rounding yielded an optimal solution. Notable exceptions are the ball and heptoroid meshes, of genus 5 and 22 respectively, where CVP did improve the solution considerably. We also show the resulting cross fields for the ball mesh with β computed by rounding (left) and CVP (right).

This is a straightforward result of Theorem 3.1 and Theorem 3.3. A solution (θ, p) is optimal for MIQ if and only if there exists a corresponding optimal solution (x, α, β) for TCO, where (α, β) are also optimal for IOQ. Thus optimality can be determined from the holonomy signature $s = [\alpha, \beta]$.

3.3 IOQr: Relaxation

We will consider a relaxation of Equation (11), with the β variables relaxed to be real-valued. Note that in this case we can always make $b(\alpha, \beta)$ equal 0 by taking $\beta^*(\alpha) = \frac{2}{\pi}\left(\beta_g(H) + H^T d_0 a(\alpha)\right)$. Thus, we can eliminate β and solve

$$\begin{aligned} \underset{\alpha \in \mathbb{Z}^n}{\text{minimize}} \quad & E_I(\alpha) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{n} \alpha_i = 4\chi. \end{aligned} \tag{12}$$

Then, given the solution α^* to the above, we further solve:

$$\underset{\beta \in \mathbb{Z}^n}{\text{minimize}} \quad \|Bb(\alpha^*, \beta)\|_2^2. \tag{13}$$

In the following two sections we first propose an iterative algorithm for minimizing the energy in Equations (12), (13) and then show how to devise an approximation that allows us to trade-off the quality of the cross field and the computational time.

4 OPTIMIZATION

4.1 Solving for α

The optimization problem in Equation (12) is an instance of the *closest vector problem* (CVP) [Micciancio 2001], known to be NP-hard in the general setting. While there exist instances of the problem that are polynomially solvable [Sahraei and Gastpar 2017], to the best of our knowledge such an algorithm is not currently known for matrices of the form of $L^\dagger$. Furthermore, our problem has an additional complication due to the sum constraint on α. We therefore opt for an *iterative* approach that has some favorable properties: (i) the constraint holds by construction, (ii) it is easily parallelizable, and (iii) it is closely related to the *resistance distance* and thus admits a graph-theoretic approximation.

Assume $\alpha^{(t)} \in \mathbb{Z}^n$ is a feasible solution for Equation (12), and consider the update $\alpha^{(t+1)} = \alpha^{(t)} + h_{ij}$, where $h_{ij} = h_i - h_j$, and $h_i \in \mathbb{Z}^n$ is a vector that is all zeros except for a single 1 at the i-th entry. Note that $\alpha^{(t+1)}$ sums to 4χ. Thus, to minimize (12), we start with a random feasible $\alpha^{(0)} \in \mathbb{Z}^n$, and iteratively update it by adding the best h_{ij} over all possible choices of i, j, $i \neq j$ as follows:

$$\begin{aligned} (i^*, j^*) &= \underset{1 \le i, j \le n,\, i \neq j}{\arg\min}\; E(\alpha^{(t)} + h_{ij}), \\ \alpha^{(t+1)} &= \alpha^{(t)} + h_{i^* j^*}. \end{aligned}$$

We continue this process as long as there exists a choice of (i, j) that reduces the energy.

While global optimality cannot be guaranteed, we have some partial guarantees since there is no h_{ij} that reduces the energy. Specifically, it is easy to see that there is no relocation of a single cone singularity of magnitude $\pm\frac{\pi}{2}$, and no cancellation of two such singularities that reduces the energy.

4.2 Solving for β

The optimization problem in Equation (13) is also a CVP, of dimension $2g$, with the matrix $\frac{\pi}{2}B(H^T B)^{-1}$ and the target vector $B(H^T B)^{-1}(\beta_g + H^T d_0 a)$. For low dimensional lattices, and low resolution meshes, i.e. when g, n are small, finding the optimal solution is still computationally feasible [Agrell et al. 2002]. However, we found that direct rounding yields excellent results, and in many cases the exact CVP solution did not considerably improve the energy. This is demonstrated in Figure 3, which shows the energy $\|Bb(\alpha, \beta)\|_2^2$ computed using direct rounding and using the exact CVP solution. Thus, in our experiments we use $\beta = \text{round}(\beta^*(\alpha^*))$. A large improvement in the energy did occur for some of the higher genus meshes, implying an interesting future research direction.

4.3 Initialization

To initialize $\alpha^{(0)}$, we pick a set of random indices $S \subseteq \{1, \ldots, n\}$ and set $\alpha^{(0)}$ at these locations to ± 1 such that $\sum_i \alpha_i^{(0)} = 4\chi$ holds.

To check the stability of our algorithm to this initialization, we ran it on the Bunny mesh with a varying number of initial singularities, $N = 30$ times for each $|S|$ value. For each run, we measured the resulting final energy, and the resulting number of cone singularities. As is evident in Figure 4, both the energy (left) and the final number of singularities (right) are stable under the choice of initial random

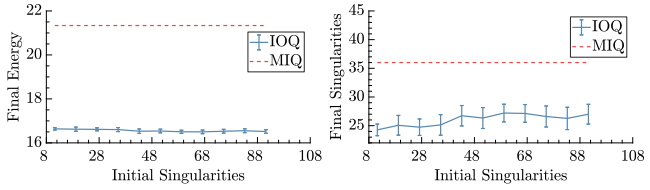

Fig. 4. Final energy value (left) and final number of cone singularities (right) as we increase the number of initial singularities. We also show the energy value and number of singularities of MIQ [Bommes et al. 2009]. Note that our method is stable to the initialization, and in all cases yields a better energy than MIQ with fewer singularities.

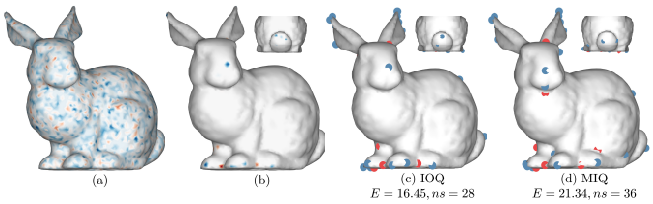

Fig. 5. The average of α across N random initializations for the initial (a) and final (b) iterations. Note that the singularities concentrate at specific locations, yet there might be multiple equivalent configurations, e.g. on the tail. We also show one of our results (c) and the MIQ result (d).

input, even when the number of initial singularities is far larger than their final number. For reference, we also show the energy value and the number of singularities of MIQ for this mesh. Note that for all runs our results yield a lower number of singularities and a lower energy.

As the figure shows, there is a larger variability in the final number of singularities than in the energy. We believe this is because there are multiple solutions that lead to similar energy values. To demonstrate this, we additionally show in Figure 5 the average of α across all N experiments for the initial (a) and final (b) iterations. Note that while the singularities concentrate at specific locations, there might be multiple equivalent configurations, e.g. on the tail, leading to a greater variation in the singularities than in the energy. We additionally show one of our results (c), and the MIQ result (d).

4.4 Convergence.

Our algorithm converges in a relatively small number of iterations. Figure 10 (left) shows the energy values during the iterations, for different numbers of initial singularities, additional to the minimal number required to fulfill the constraints on α. Note that starting from more singularities leads to slower conversion; thus, in our experiments we always use for initialization the minimal number possible of $\pm\frac{\pi}{2}$ singularities.

4.5 Parallelization

Our iterative update for α is computationally demanding, yet easily parallelizable. To devise a practical algorithm, we note the change in energy due to the addition of h_{ij}:

$$\frac{\pi^2}{4}E_I(\alpha + h_{ij}) = \frac{\pi^2}{4}\left(E_I(\alpha) + h_{ij}^T u(\alpha) + R_{ij}\right),$$

where

$$R_{ij} = h_{ij}^T L^\dagger h_{ij}, \quad u(\alpha) = 2L^\dagger(\alpha - \tfrac{2}{\pi}\alpha_g).$$

Note that the matrix R, whose (i,j)-th entries are given by R_{ij}, is *independent* of α and can be precomputed. Furthermore, we have:

$$u(\alpha + h_{ij}) = 2L^\dagger(\alpha - \tfrac{2}{\pi}\alpha_g + h_{ij}) = u(\alpha) + 2L^\dagger h_{ij}.$$

Thus we can compute $u^{(0)} = u(\alpha^{(0)})$ and update it at every iteration. Hence, at each iteration we do the following:

$$(i^*, j^*) = \underset{1\le i,j\le n,\, i\neq j}{\arg\min}\; u_i^{(t)} - u_j^{(t)} + R_{ij},$$
$$\alpha^{(t+1)} = \alpha^{(t)} + h_{i^*} - h_{j^*},$$
$$u^{(t+1)} = u^{(t)} + 2L_{i^*}^\dagger - 2L_{j^*}^\dagger,$$

where $L_i^\dagger$ is the i-th column of $L^\dagger$. The resulting algorithm, denoted IOQr, is given in Algorithm 1.

The algorithm is highly parallel, since the computation for every (i,j) is independent, and thus it is naturally amenable to a GPU implementation. It does, however, require the precomputation of $L^\dagger$, which may be prohibitive for large meshes. In our experiments, this algorithm was adequate for meshes with up to $40K$ faces (see Figure 3 in the supplemental material) on an NVIDIA GeForce GTX 1080 Ti GPU. For larger meshes, we propose an approximation scheme described in the next section.

5 APPROXIMATION

5.1 Background: The Resistance Distance

The matrix R that appears in Algorithm 1 has a geometric meaning, and is known as the *graph resistance distance* matrix. It encodes graph-theoretical distances originally used in the theory of electrical networks [Kirchhoff 1958]. On an edge, it is equal to the potential difference when we inject a unit current at one end of the edge and extract it at the other end. It can also be thought of as the commute time between two vertices [Chandra et al. 1996], or the probability that an edge appears in a random spanning tree of the graph [Bollobás 2013; Doyle and Snell 1984]. The resistance distance, also known as the *commute time distance*, has been used in geometry processing for various applications; see e.g. [Patané and Spagnuolo 2013]. It can be computed explicitly by:

$$R_{ij} = h_{ij}^T L^\dagger h_{ij} = L_{ii}^\dagger + L_{jj}^\dagger - 2L_{ij}^\dagger,$$

ALGORITHM 1: IOQr, IOQϵ

input : $\mathcal{V}$, $\mathcal{F}$, $\mathcal{E}$, d_0, H, $L^\dagger$, R, $\tilde{R}_\epsilon$
output: α, β
n, m, $l \leftarrow$ number of vertices, faces, and edges respectively;
$\alpha^{(0)} \leftarrow$ random placement of ± 1 such that $\sum_i \alpha_i^{(0)} = 4\chi$, $\alpha^{(0)} \in \mathbb{Z}^n$;
$m^{(0)} \leftarrow -\infty$; $t \leftarrow 0$;
$u^{(0)} \leftarrow 2L^\dagger(\alpha^{(0)} - \frac{2}{\pi}\alpha_g)$; $u^{(0)} = \text{solve}(L, 2\alpha^{(0)} - \frac{4}{\pi}\alpha_g)$;
while $m^{(t)} < 0$ **do**
  $i, j \leftarrow \underset{1\le i,j\le n,\, i\neq j}{\arg\min}\; u_i^{(t)} - u_j^{(t)} + R_{ij}$; ... $+(\tilde{R}_\epsilon)_{ij}$;
  $\alpha^{(t+1)} \leftarrow \alpha^{(t)} + h_i - h_j$;
  $u^{(t+1)} \leftarrow u^{(t)} + 2L_i^\dagger - 2L_j^\dagger$; $u^{(t+1)} = \text{solve}(L, 2\alpha^{(t+1)} - \frac{4}{\pi}\alpha_g)$;
  $m^{(t+1)} \leftarrow u_i^{(t)} - u_j^{(t)} + R_{ij}$; ... $+(\tilde{R}_\epsilon)_{ij}$;
  $t \leftarrow t + 1$;
end
$a = \text{solve}(L, \frac{\pi}{2}\alpha^{(t)} - \alpha_g)$;
$\beta = \text{round}\left(\frac{2}{\pi}(\beta_g(H) + H^T d_0 a)\right)$;

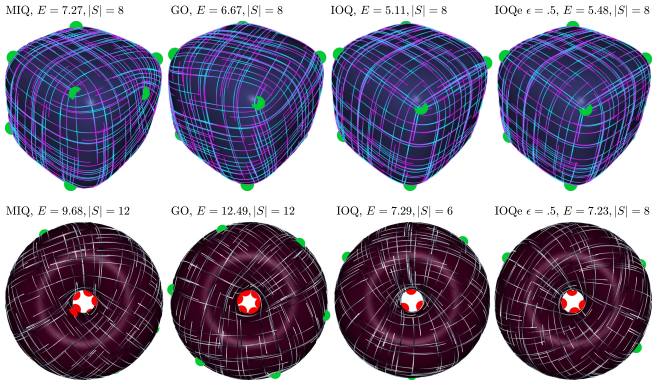

Fig. 6. The results of our exact and approximated algorithms (IOQ and IOQe), compared with MIQ [Bommes et al. 2009] and GO [Knöppel et al. 2013], on two symmetric models. See the text for details.

yet this requires the matrix $L^\dagger$, which is computationally prohibitive to compute for large meshes. To overcome this, we use an efficient random approximation of R.

5.2 Random Approximation

Spielman and Srivastava [2011] proposed a method, based on random projections, to approximate the resistance distance of a given graph. Calculating R is equivalent to computing the pair-wise Euclidean distances between points in $\left\{d_0 L^\dagger h_i\right\}_{v_i \in \mathcal{V}}$, which are the columns of $d_0 L^\dagger$. This is true since

$$\begin{aligned} R_{ij} &= h_{ij}^T L^\dagger h_{ij} = h_{ij}^T L^\dagger L L^\dagger h_{ij} \\ &= \left(h_{ij}^T L^\dagger d_0^T\right)\left(d_0 L^\dagger h_{ij}\right) = \|d_0 L^\dagger h_{ij}\|_2^2. \end{aligned}$$

To efficiently and accurately approximate these, Spielman and Srivastava [2011] projected the columns of $d_0 L^\dagger$ onto a subspace spanned by $O(\log n)$ random vectors and calculate the distances in the *projected space*. Let $Q_{k\times l}$ be a random *Bernoulli* matrix with entries of $\pm 1/\sqrt{k}$, where $k \geq 24 \log n/\epsilon^2$, for some $\epsilon > 0$. They computed $Y = Qd_0$ and solve $ZL = Y$ for Z. The n columns of Z are the projected points of dimension k, and we can use the Euclidean distances between them to approximate the resistance distance by:

$$(\tilde{R}_\epsilon)_{ij} = \|Z_i - Z_j\|_2^2,$$

where Z_i is the i-th column of Z. The algorithm for computing $\tilde{R}_\epsilon$ is provided in Algorithm 2.

ALGORITHM 2: Approximate resistance distance

input : $\mathcal{V}$, $\mathcal{E}$, ϵ
output: $\tilde{R}_\epsilon$
$k \leftarrow \text{round}(24 \log n/\epsilon^2)$;
$Q \leftarrow$ random $k \times l$ matrix with entries $\pm 1/\sqrt{k}$;
$Y \leftarrow Qd_0$;
$Z^T = \text{solve}(L, Y^T)$;
$(\tilde{R}_\epsilon)_{ij} = \|Z_i - Z_j\|_2^2$;

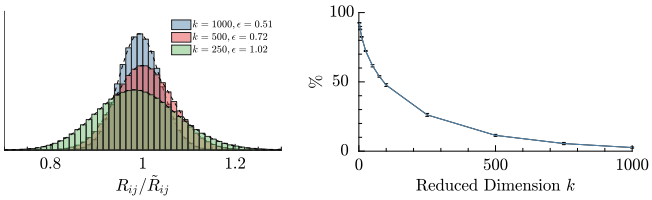

Fig. 7. Left: Histogram of the pair-wise distortions $R_{ij}/(\tilde{R}_\epsilon)_{ij}$ for three values of ϵ. Right: Percentage of pairs with distortion greater than 10% as we increase the projected dimension k. See the text for details.

Implementation. To compute Z we first factorize L, and then use back-substitution, using [Davis 2013]. The pair-wise distances between all the columns in Z are then computed efficiently on the GPU using the built-in Matlab function "pdist". The algorithmic modifications required in order to work with the approximate resistances instead of R and $L^\dagger$ are minor. The only difference is that instead of using $L^\dagger$ for updating u, we need to solve a sparse linear system at every iteration, yet we can do that efficiently using the factorization of L. The changes are highlighted in Algorithm 1, where green lines replace blue lines when using approximate resistances. Note that when using approximate resistances, the energy is no longer guaranteed to monotonically decrease; thus, we stop when we identify a cycle in the solutions.

Figure 10 shows the energy values during the iterations for the approximated and exact algorithms with the same initialization. Note that the graphs are almost indistinguishable. Indeed, in practice the approximated algorithm yields energy values that are very close to the result of the exact algorithm. Figure 6 shows the output of the approximated algorithm with $\epsilon = 0.5$ compared with MIQ [Bommes et al. 2009], GO [Knöppel et al. 2013] and our exact algorithm IOQ, on two symmetric models. Note that compared to MIQ and GO on the torus, IOQ finds a lower energy solution with fewer singularities. Furthermore, for both the cube and the torus, the singularities' locations are close to symmetric. IOQe yields very similar results to IOQ, at the expense of a slightly higher energy. Note that the GO result on the cube is very similar to ours, up to a global rotation of all the crosses, which does not affect the energy.

Approximation quality. The Johnson-Lindenstrauss Lemma guarantees that with high probability the Euclidean distances will not be distorted by more than a multiplicative factor of $1 \pm \epsilon$ [Achlioptas 2001; Dasgupta and Gupta 2003; Johnson and Lindenstrauss

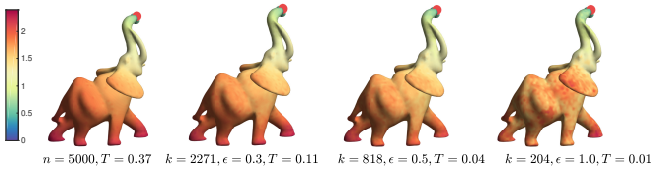

Fig. 8. The resistance distance and its approximations as we increase ϵ, with respect to the red point. Here we show the running time in seconds, T, and the original and projected dimension, n and k respectively.

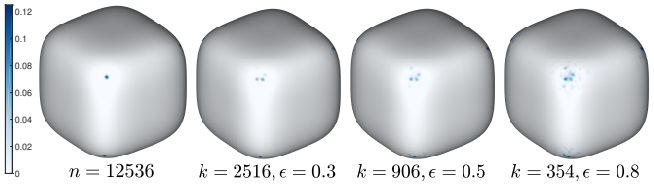

Fig. 9. Time/quality tradeoff of IOQϵ with different ϵ values. We show the average across $N = 300$ experiments of α after convergence, using (a) the exact resistance distance with different initial random α; and (b-d) the approximate resistance, with different random projections and the same initial α. Note that for $\epsilon = 0.5$ we get a dimensionality reduction of more than 90%, yet the algorithm yields excellent cone positions.

1984]. In practice, we found that for graphs that come from triangle meshes, the distortion is empirically less than the theoretical guarantee. Figure 7 (left) shows the distortion ratio of $R_{ij}/(\tilde{R}_\epsilon)_{ij}$ for different values of ϵ. For example, for $n = 55{,}000$ and $\epsilon \approx 0.5$, the reduced dimension is $k = 1000$. In this case, the expected distortion is 50%, yet the actual observed maximal distortion is only around 15%. In Figure 7 (right) we show the trend for a growing k, i.e. a decreasing ϵ, of the percentage of pairs with distortion higher than 10%, again for $n = 55{,}000$. Note that the distortion quickly decreases, with less that 5% of such pairs for $k = 1000$.

To give some insight into the behavior of the approximation, we visualize in Figure 8 the exact resistance distance R, and its approximation $\tilde{R}_\epsilon$ for different values of ϵ. We show these as color coded functions, where the function is the resistance distance of all mesh vertices from a single marked vertex. Note that, as expected, the functions become noisier as ϵ grows, yet qualitatively, they are similar to the exact resistance distance.

Time/quality tradeoff. Fig. 9 shows the robustness of this approximation when combined with our algorithm, and the resulting tradeoff between the quality of the output and the computational time. We first compute $N = 300$ different $\tilde{R}_\epsilon$ for various ϵ values, by running Algorithm 2. Then, for a fixed initial $\alpha^{(0)}$, we run Algorithm 1 with the different $\tilde{R}_\epsilon$. We show in (b-d) the average optimal α after convergence, where we average across the N experiments for matrices with the same ϵ. In addition, we show in (a) the average optimal α, when using the exact resistance distance, and starting from N random initializations, for comparison. As the figure shows, while the locations of the singularities degrade for very large ϵ, for $\epsilon = 0.5$ we achieve a dimensionality reduction of more than 90%, and still get an excellent distribution of the locations of the singularities. In all our experiments, unless mentioned otherwise, we used $\epsilon = 0.5$.

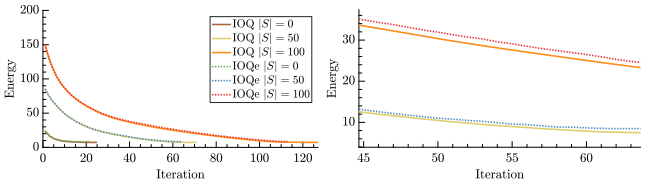

Fig. 10. Left: Energy values during the iterations for different numbers of initial singularities, for both the exact (IOQ) and approximated (IOQe) methods. Right: Close-up of the same graph. Note that the energy plot of the approximated method is nearly indistinguishable from the exact method.

6 IMPLEMENTATION

We implemented our algorithm in Matlab, using its built-in support for GPU parallelization with "gpuArray". For the inner loop that minimizes over all (i, j) pairs, we used a CUDA kernel. For the exact algorithm, we first attempt to compute $L^\dagger$ on the GPU, and if that fails due to memory requirements, we attempt a blockwise GPU inversion. If that also fails, we invert $L^\dagger$ on the CPU. For large meshes, where L cannot be inverted on the GPU, the exact algorithm is therefore computationally very expensive. For the approximate algorithm, we factorize L on the CPU, and use the factorization to compute Z. Then $\tilde{R}_\epsilon$ is computed from Z on the GPU. On our machine, with an NVIDIA GeForce GTX 1080 Ti GPU and an Intel Core i7-7820X CPU @ 3.60GHz with 8 cores, the exact algorithm takes a few minutes for a mesh with $50K$ faces, and the approximate algorithm with $\epsilon = 0.5$ takes around 20 seconds. The full timing details are provided in Figure 3 of the supplemental material.

6.1 Limitations

The main limitation of our algorithm is the heavy computational load. Because we have to fit $L^\dagger$ or R on the GPU, the memory requirement is $O(n^2/2)$, which on our hardware (12 GB RAM) was feasible for meshes with up to $100K$ faces. This could potentially be reduced to $O(nlog(n))$ by holding Z instead on the GPU and computing $\tilde{R}_\epsilon$ on the fly.

Another limitation is that we do not handle directional constraints. In many scenarios, especially quadrangular remeshing, it is beneficial if the cross field is aligned with the curvature directions of the surface. We leave the generalization of our approach to directional constraints for future work.

Finally, the geometry of the surface is not incorporated in the system matrix L, since we use $L = d_0^T d_0$, and not a weighted Laplacian. The TC formulation allows for adding weights on the edges, [Crane

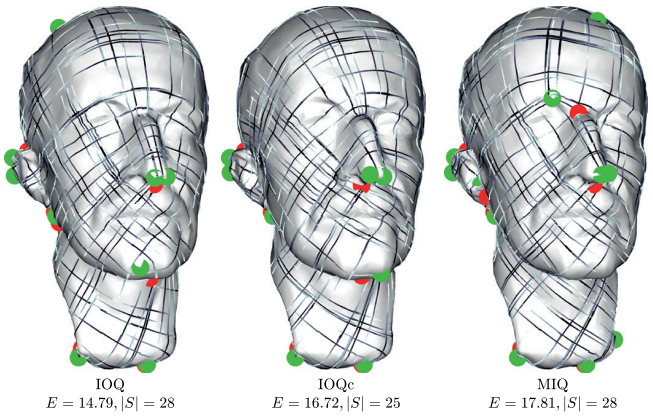

Fig. 11. An example of our method with the graph Laplacian (left) and the cotangent Laplacian (middle) compared to MIQ (right). Note that in both cases our method finds a better solution in terms of smoothness energy and singularity placement.

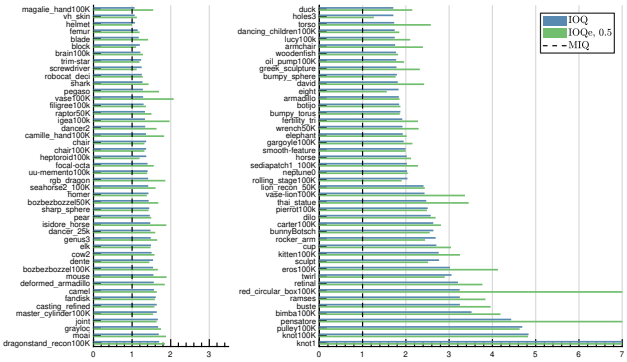

Fig. 12. The improvement in energy and singularities of our exact (IOQ) and approximate (IOQe) methods relative to MIQ, computed by $(E_{MIQ}/E_{ours})((|S|_{MIQ}+1)/(|S|_{ours}+1))$ on the meshes from [Myles et al. 2014], where $|S|$ is the number of singularities. Note that for all meshes our result is bigger than 1, and thus improves on MIQ. Furthermore, the results of IOQ and IOQe are comparable for most meshes.

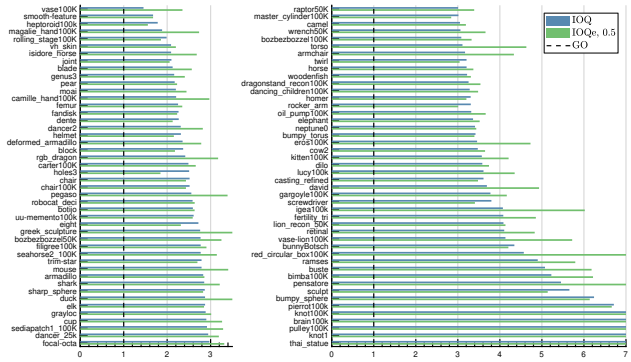

Fig. 13. The improvement in energy and singularities of our exact (IOQ) and approximate (IOQe) methods relative to GO [Knöppel et al. 2013], using the same protocol as in Figure 12. Here, again, for all meshes our result is bigger than 1, and thus improves on GO. Furthermore, the results of IOQ and IOQe are comparable for most meshes.

et al. 2010, Section 2.5], and a similar approach could be applied to MIQ as well. We did not attempt that, as we wished to compare to a non-modified MIQ. As the approach of Spielman and Srivastava [2011] can be applied to a weighted graph, we believe a suitable adaptation of our algorithm that incorporates the geometry in L can be devised. See Figure 11 for a preliminary result in which we replaced L by the cotangent Laplacian and no approximation was used. Note that the resulting cross-field has fewer singularities and a lower smoothness energy than the MIQ result.

7 RESULTS

7.1 Comparisons

Quantitative, with MIQ [Bommes et al. 2009]. We compared our exact method (IOQ), our approximated method (IOQe with $\epsilon = 0.5$) and MIQ [Bommes et al. 2009]. For IOQ, IOQe and MIQ we used the same single directional constraint on an arbitrary face. GO can be computed without any directional constraints. To evaluate the methods, we assessed the angle-based energy of the output fields $E = E_M(\theta, p)$ from Equation (2), the number of singularities $|S|$, i.e. the number of vertices $v_i \in \mathcal{V}$ such that $\alpha_i \neq 0$, and the timing, on the models from the benchmark provided by [Myles et al. 2014]. We implemented our method in Matlab and CUDA, while for MIQ we used the Libigl [Jacobson et al. 2016] implementation.

Figure 12 shows the ratio of improvement of our methods vs. the MIQ result, i.e. $(E_{MIQ}/E_{ours})((|S|_{MIQ}+1)/(|S|_{ours}+1))$. As the figure shows, the product of these ratios is always greater than 1 (the vertical black line); thus, for all models we improve upon MIQ. Also note that the approximate method (IOQe) yields comparable, and sometimes better, results than IOQ. The median improvement ratio over all models is 1.71 for IOQ and 1.86 for IOQe.

Quantitative, with GO [Knöppel et al. 2013]. The cross field generation method suggested by Knöppel et al. [2013] optimizes a different energy than MIQ, yet is very efficient and has global optimality guarantees (for their energy). We compared our method with GO as well, using a Matlab implementation with a face-based discretization as done in [Diamanti et al. 2014]. The results are shown in Figure 13, using the same quantitative measures as in Figure 12. Since we are measuring an energy that GO is not optimizing for, our method outperforms GO in terms of energy. Note, however, that we also outperform GO in terms of the number of singularities. The median improvement rate over all models was 2.99 for IOQ and 3.3 for IOQe. All the quantitative results, i.e. E, $|S|$ for all the methods and all the models are provided in Figures 1 and 2 in the supplemental material.

Timing. Our approach is considerably slower than both MIQ and GO. For example, MIQ completed for all the models within a few seconds, whereas GO is even faster, finishing in less than a second for all models. Our approximated method with $\epsilon = 0.5$ takes around 20 seconds on models of size $50K$ faces. Our exact method can take around 5 minutes for such models and around 20 minutes for models of $100K$ faces. However, it is efficient for models whose L matrix can be inverted on the GPU. All the timing results for IOQ and IOQe are provided in Figure 3 in the supplemental material.

Qualitative. Figure 15 shows a few example models from the benchmark with their corresponding cross fields for the different methods. Note that for the IOQ approach there are considerably fewer singularities, with a smaller energy, and the singularities are well distributed when compared with the MIQ and GO results.

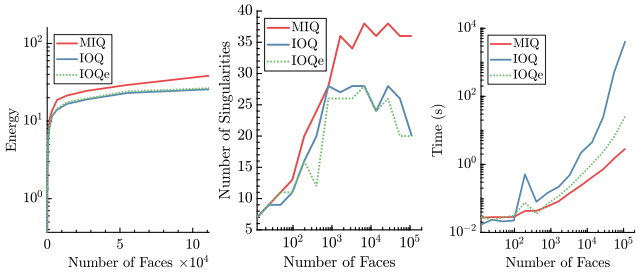

Fig. 14. Energy, number of singularities and timing scalability, on a series of meshes with varying resolutions. See the text for details.

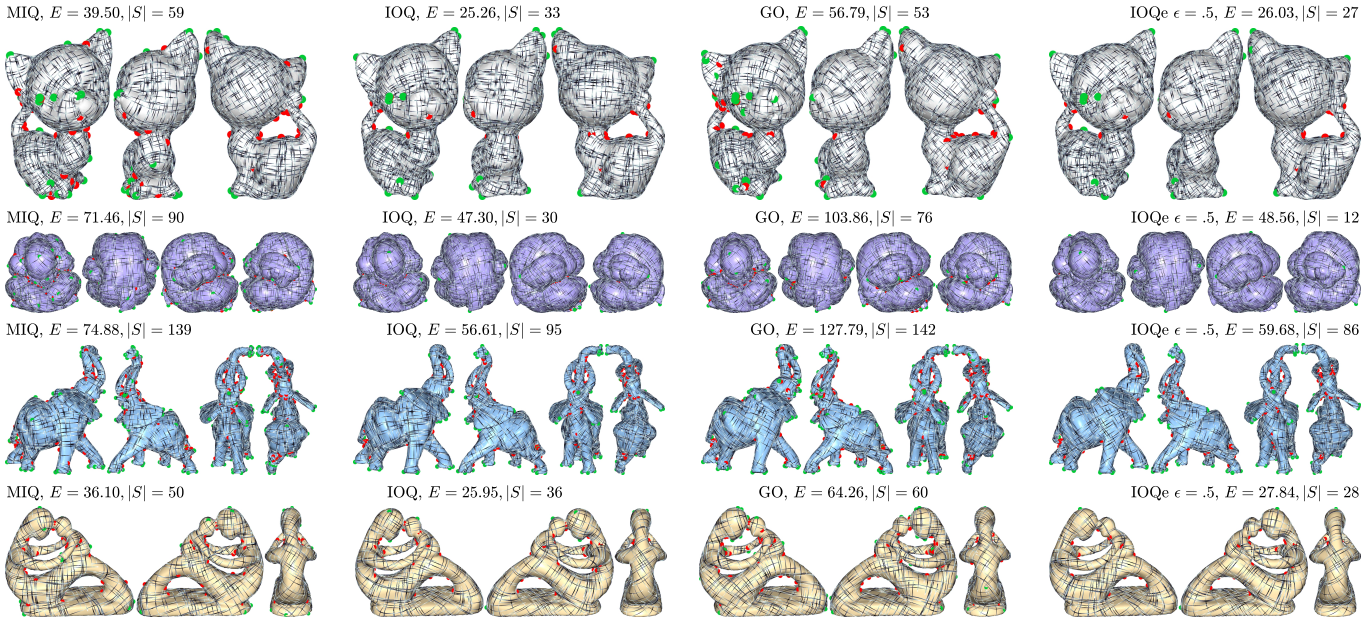

Fig. 15. Some meshes from the benchmark and their cross fields. Note that our method yields considerably fewer singularities with lower energy values.

7.2 Scalability

To compare the scalability of our approach with respect to the energy, number of singularities and timing, we ran our methods and MIQ on a series of meshes with the same geometry and varying resolution of the Bunny mesh. As Figure 14 shows, our energy remains the same order of magnitude when increasing the mesh size, and the number of singularities remains largely the same. In terms of timing, both IOQ and IOQe are slower than MIQ, yet IOQe is considerably faster than IOQ.

7.3 Varying ϵ

Figure 16 shows our results on a model with sharp features as we increase ϵ, the approximation parameter. Note that with $\epsilon = 1$, which reduces the dimension to a mere 3% of the original dimension, IOQe still finds a similar singularity placement, albeit with a somewhat higher energy. For the best trade-off between the resulting smoothness energy and the reduced dimension, we have found that taking $\epsilon = 0.5$ is appropriate.

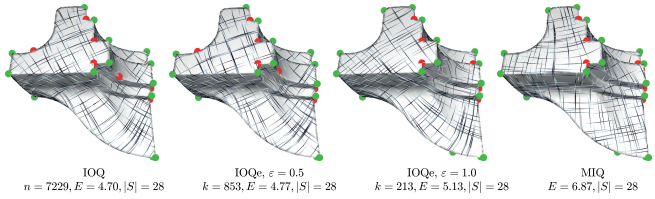

Fig. 16. Varying the approximation parameter, ϵ, has little effect on the final singularity placement. Note in particular that even with a projected dimension of $k = 213$, our method still places most of the singularities on the corners as desired, albeit with a somewhat higher energy.

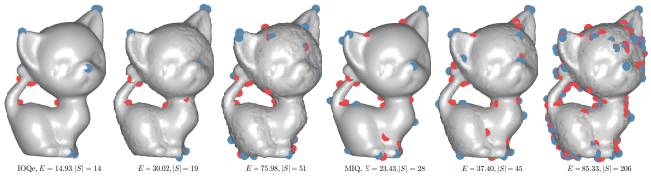

Fig. 17. Stability of our method (left) compared to MIQ (right) with increasing levels of normal noise added to the vertex positions.

7.4 Stability to Noise

To measure the stability of our approach to geometric noise, we added random Gaussian noise in the normal direction to the kitten model, with standard deviation of 10 and 15 percent of the average edge length. The results are shown in Figure 17, for IOQe with $\epsilon = 0.5$ (first three from the left) and MIQ (last three from the right). Note that for the 10 percent noise level the number of singularities remains almost the same for our approach. Even for the higher

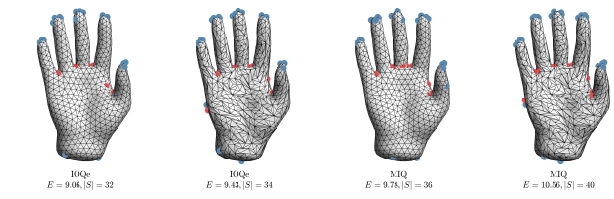

Fig. 18. The results of our method (left) compared to MIQ (right) on a uniform and non-uniform triangulation. Note that while our method does not incorporate the geometry in the system matrix L, it is, at least to some extent, robust to changes in the triangulation.

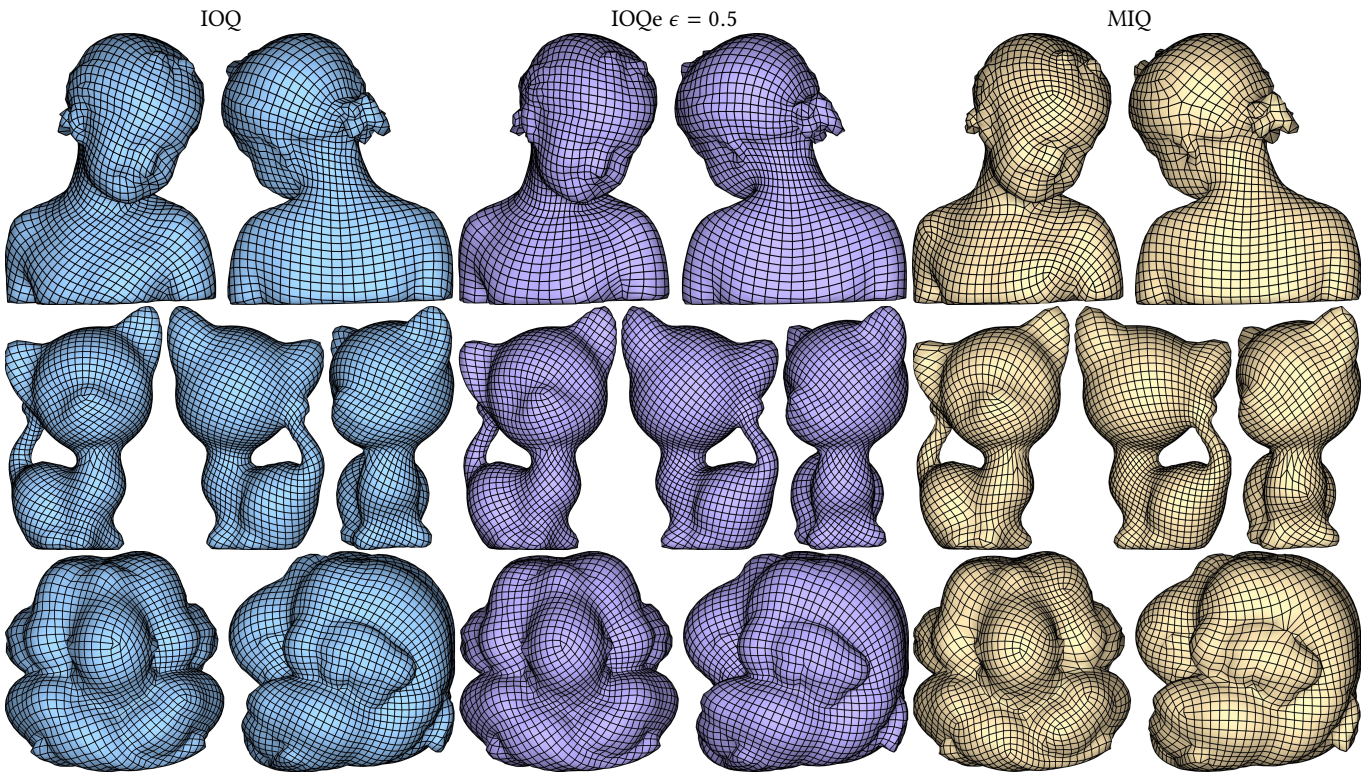

Fig. 19. Quadrangular meshes generated from our cross fields (left,center), compared to quad meshes generated from MIQ cross fields (right).

noise level, our algorithm yields considerably fewer singularities than MIQ, as well as a lower energy.

In another experiment, we randomly flipped the edges of a mesh to get a non-uniform triangulation. While the geometry is not encoded in the system matrix L, it does contribute to the right hand side of the system through the angle defects α_g, β_g. Therefore, the results are, at least to some extent, robust to changes in the triangulation, as is shown in Figure 18.

7.5 Application to Quadrangulation

The cross fields we generate can be used for computing quadrangular meshes, by creating a parameterization whose gradients align with the cross directions, and then extracting the quads. We used off-the-shelf approaches for these steps: the parameterization part of MIQ [Bommes et al. 2009, Sec. 5] as implemented in libigl [Jacobson et al. 2016], and the quad extraction method QEx [Ebke et al. 2013] that receives as input a parameterization. While in general curvature direction alignment is often required for quad meshing, it seems that in some cases good cone point locations can lead to high quality quadrangular meshes even without alignment. Figure 19 shows some examples of quad meshes computed using our cross fields, and using the cross field generated by MIQ. Note that the better placed singularities and lower energy of our approach lead to smoother, more symmetric quad meshes with better shaped quads.

8 CONCLUSION

We showed an equivalence between two existing methods for generating cross fields, which we then used to formulate a new iterative algorithm that finds better solutions than state-of-the-art results in terms of the angle-based energy and number of singularities. Using a recent approximation of the resistance distance, based on random projections, we allow a trade-off between the computation time and the smoothness of the produced field using a single tunable parameter, ϵ.

A natural generalization of our approach is to add support for directional constraints. In addition, we believe that a similar iterative approach could be useful for computing a parameterization with integer cone locations that is aligned with a given cross field. Finally, to the best of our knowledge, this approximation of the resistance distance has not been used in geometry processing, but we posit that it is of independent interest and could be useful in other applications.

ACKNOWLEDGMENTS

The work was supported by the European Research Council (ERC starting grant No. 714776 “OPREP”), and the Israel Science Foundation (grant No. 504/16). We thank AIM@SHAPE, Stanford, Keenan Crane and Ashish Myles for providing the models.

REFERENCES

Dimitris Achlioptas. 2001. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database*

systems. ACM, 274–281.
Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger. 2002. Closest point search in lattices. *IEEE transactions on information theory* 48, 8 (2002), 2201–2214.
Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 449–458.
Béla Bollobás. 2013. *Modern graph theory*. Vol. 184. Springer Science & Business Media.
David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-integer quadrangulation. *ACM Transactions On Graphics (TOG)* 28, 3 (2009), 77.
Alon Bright, Edward Chien, and Ofir Weber. 2017. Harmonic global parametrization with rational holonomy. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 89.
Marcel Campen and Denis Zorin. 2017a. On Discrete Conformal Seamless Similarity Maps. *arXiv preprint arXiv:1705.02422* (2017).
Marcel Campen and Denis Zorin. 2017b. Similarity Maps and Field-Guided T-Splines: a Perfect Couple. *ACM Trans. Graph* 36, 4 (2017).
Ashok K Chandra, Prabhakar Raghavan, Walter L Ruzzo, Roman Smolensky, and Prasoon Tiwari. 1996. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity* 6, 4 (1996), 312–340.
Keenan Crane, Fernando De Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*. ACM, 7.
Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial connections on discrete surfaces. In *Computer Graphics Forum*, Vol. 29. 1525–1533.
Sanjoy Dasgupta and Anupam Gupta. 2003. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms* 22, 1 (2003), 60–65.
Timothy A Davis. 2013. Algorithm 930: FACTORIZE: An object-oriented linear system solver for MATLAB. *ACM Transactions on Mathematical Software (TOMS)* 39, 4 (2013), 28.
Olga Diamanti, Amir Vaxman, Daniele Panozzo, and Olga Sorkine-Hornung. 2014. Designing N-PolyVector Fields with Complex Polynomials. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 1–11.
Peter G Doyle and J Laurie Snell. 1984. *Random walks and electric networks*. Mathematical Association of America,.
Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. 2013. QEx: robust quad mesh extraction. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 168.
David Eppstein. 2003. Dynamic generators of topologically embedded graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 599–608.
Alec Jacobson, Daniele Panozzo, et al. 2016. libigl: A simple C++ geometry processing library. (2016). http://libigl.github.io/libigl/.
Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant field-aligned meshes. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 189.
William B Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics* 26, 189-206 (1984), 1.
G Kirchhoff. 1958. On the solution of the equations obtained from the investigation of the linear distribution of galvanic currents. *IRE transactions on circuit theory* 5, 1 (1958), 4–7.
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 59.
Christian Liebchen and Romeo Rizzi. 2007. Classes of cycle bases. *Discrete Applied Mathematics* 155, 3 (2007), 337–355.
Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 35–57.
Daniele Micciancio. 2001. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory* 47, 3 (2001), 1212–1215.
Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 135.
Ashish Myles and Denis Zorin. 2012. Global parametrization by incremental flattening. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 109.
Ashish Myles and Denis Zorin. 2013. Controlled-distortion constrained global parametrization. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 105.
Giuseppe Patané and Michela Spagnuolo. 2013. An interactive analysis of harmonic and diffusion equations on discrete 3D shapes. *Computers & Graphics* 37, 5 (2013), 526–538.
Chi-Han Peng, Eugene Zhang, Yoshihiro Kobayashi, and Peter Wonka. 2011. Connectivity editing for quadrilateral meshes. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 141.
Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic global parameterization. *ACM Transactions on Graphics (TOG)* 25, 4 (2006), 1460–1485.
Nicolas Ray, Bruno Vallet, Wan Chiu Li, and Bruno Lévy. 2008. N-symmetry direction field design. *ACM Transactions on Graphics (TOG)* 27, 2 (2008), 10.
Saeid Sahraei and Michael Gastpar. 2017. Polynomially solvable instances of the shortest and closest vector problems with applications to compute-and-forward. *IEEE Transactions on Information Theory* 63, 12 (2017), 7780–7792.
Daniel A Spielman and Nikhil Srivastava. 2011. Graph sparsification by effective resistances. *SIAM J. Comput.* 40, 6 (2011), 1913–1926.
Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal equivalence of triangle meshes. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 77.
Yiying Tong, Santiago Lombeyda, Anil N Hirani, and Mathieu Desbrun. 2003. Discrete multiscale vector field decomposition. In *ACM transactions on graphics (TOG)*, Vol. 22. ACM, 445–452.
Amir Vaxman, Marcel Campen, Olga Diamanti, Daniele Panozzo, David Bommes, Klaus Hildebrandt, and Mirela Ben-Chen. 2016. Directional Field Synthesis, Design, and Processing. *Computer Graphics Forum* (2016).

APPENDIX - PROOFS.

Let $H \in \mathbb{Z}^{l\times 2g}$ be any integral basis for the non-contractible dual cycles of $\mathcal{M}$, e.g. as obtained by computing the tree-cotree decomposition [Eppstein 2003, Lemma 3.2] and orienting the edges [Crane et al. 2010, Sec 2.1], and let $d_1 \in \mathbb{Z}^{m\times l}$ be the signed face-edge adjacency matrix. Further, let $\mathcal{T} \subset \mathcal{E}^*$ be a dual spanning tree of $\mathcal{M}$.

First, we relate an MIQ solution (θ, p) and a TC solution (x, α, β) through the following system of linear equations:

$$\frac{\pi}{2}(\Gamma p + s) = s_g(\Gamma) - \Gamma r, \tag{14}$$

$$\frac{\pi}{2} p + d_1^T \theta + x = -r, \tag{15}$$

where $\Gamma^T = \left[d_0, H\right]$, $s = \left[\alpha, \beta\right]$, and $s_g = s_g(\Gamma) = \left[\alpha_g, \beta_g(H)\right]$.

Lemma 8.1. *For any choice of H, we have that $I_0(\Gamma) = \frac{2}{\pi}(s_g(\Gamma) - \Gamma r) \in \mathbb{Z}$, and $\sum_{i=1}^n I_0(\Gamma)(i) = 4\chi$.*

Proof. By definition, $s_g(\Gamma)$ are the geometric angle defects around the cycles of Γ, and Γr is the *holonomy* of the local reference frame field along the cycles of Γ (see, e.g. [Crane et al. 2010, Sec. 3.2]). Hence, their difference is a multiple integer of $\frac{\pi}{2}$ and thus $I_0 \in \mathbb{Z}$. Furthermore, for $i = 1..n$ we have that $I_0(i)$ is the index of the reference frame field at the vertex $v_i \in \mathcal{V}$, and thus $\sum_{i=1}^n I_0(\Gamma)(i) = 4\chi$ [Ray et al. 2008, Theorem 2.3].

Lemma 8.2. *Given a dual spanning tree $\mathcal{T} \subset \mathcal{E}^*$, let Γ_f be the matrix formed by the columns of Γ corresponding to $\mathcal{E}^* \setminus \mathcal{T}$, i.e. dual edges not in $\mathcal{T}$. In addition, let $s \in \mathbb{Z}^{n+2g}$ such that $\sum_{i=1}^n s_i = 4\chi$. Then the linear system*

$$\Gamma_f p_f = -s + I_0(\Gamma) \tag{16}$$

has a unique integer solution.

Proof. Every column of d_0^T sums to 0, thus we can remove its first row and it will still span all the dual cycles. Given a matrix M, let $\hat{M}$ denote the matrix with the first row removed. Hence, $\hat{\Gamma}$ is an integral basis for the dual cycles, with independent rows. Using [Liebchen and Rizzi 2007, Lemma 27], we have that the matrix $\hat{\Gamma}_f$ is a non-singular unimodular matrix. Furthermore, the right hand side of Equation (16) is integer, since s is integer and due to Lemma 8.1. Thus the system $\hat{\Gamma}_f p_f = -\hat{s} + I_0(\hat{\Gamma})$ has a unique integer solution. Finally, we have that $\sum_{i=1}^n I_0(\Gamma)(i) = 4\chi$ from Lemma 8.1, and $\sum_{i=1}^n s_i = 4\chi$. Thus $\sum_{i=1}^n (-s_i + I_0(\Gamma)(i)) = 0$. Furthermore, $\sum_{i=1}^n \Gamma_f(i,:) p_f = 0$, thus, p_f fulfills the first linear constraint as well, and is thus a unique integer solution of Equation (16) as required.

Lemma 8.3. *Let (θ, p) be a feasible solution of Equation* (2) *and let H be an integral basis of non-contractible dual cycles. Then there exists a unique solution (x, s) of the system of equations* (14), (15) *such that $s = \left[\alpha, \beta\right]$ and (x, α, β) is feasible for* (5).

Proof. Let $s = I_0(\Gamma) - \Gamma p$, then Equation (14) holds for (s, p). Note that $I_0(\Gamma)$ is integer due to Lemma 8.1, and Γp is integer since Γ and p are both integer. Thus, s is integer. Let $\lceil \alpha, \beta \rfloor = s$, with $\alpha \in \mathbb{Z}^n$, $\beta \in \mathbb{Z}^{2g}$. Then, $\alpha = I_0(d_0^T) - d_0^T p$, namely α is a vector containing the indices of the cross field (θ, p) [Bommes et al. 2009, Sec 4.1], and thus $\sum_{i=1}^n \alpha_i = 4\chi$ [Ray et al. 2008, Theorem 2.3].

Let $x = -r - \frac{\pi}{2} p - d_1^T \theta$, then Equation (15) holds for (x, θ, p). Now, we have that $\Gamma x = -\Gamma r - \frac{\pi}{2}\Gamma p - \Gamma d_1^T \theta$. Note that $d_1 \Gamma^T = 0$, since $d_1 d_0 = 0$, and the columns of H are *closed discrete one-forms* (see e.g. [Crane et al. 2013, Sec. 8.2.2]). From Equation (14) we have that $-\Gamma r - \frac{\pi}{2}\Gamma p = \frac{\pi}{2} s - s_g(\Gamma)$, thus $\Gamma x - \frac{\pi}{2} s = -s_g(\Gamma)$ and (x, s) is feasible for the problem (5).

Lemma 8.4. *Let (x, α, β) be a feasible solution of Equation (5) with cycle basis H, and let $\mathcal{T}$ be a dual spanning tree. Then there exists a unique solution (p, θ) of the system of equations (14) and (15) such that (p, θ) is feasible for (2).*

Proof. Let $s = \lceil \alpha, \beta \rfloor$. Since (x, α, β) is feasible for Equation (5), $\sum_{i=1}^n s_i = 4\chi$. Now, let p_f be the unique integer solution of Equation (16), guaranteed by Lemma 8.2. Define the vector $p \in \mathbb{Z}^l$, such that $p(e) = 0, \forall e \in \mathcal{T}$ and $p(e) = p_f(e), \forall e \in \mathcal{E}^* \setminus \mathcal{T}$. Then, p is the unique integer solution to Equation (14) that is also feasible for Equation (2). Let $\omega = -r - \frac{\pi}{2} p - x$, then ω has a unique decomposition (up to two constants) as $\omega = d_0 \omega_0 + d_1^T \omega_2 + B\omega_1$. Since (x, α, β) is feasible for TC, we have that $\Gamma x = \frac{\pi}{2} s - s_g$. Furthermore, we have that $\frac{\pi}{2} s - s_g = -\Gamma r - \frac{\pi}{2}\Gamma p$, due to Equation (14) . Hence $-\Gamma x - \Gamma r - \frac{\pi}{2}\Gamma p = \Gamma\omega = 0$, and thus ω is in the kernel of d_0^T and H^T. Since $B^T = H^T - H^T d_0 (d_0^T d_0)^\dagger d_0^T$, we have that ω is also in the kernel of B, and thus ω is orthogonal to the image of d_0 and the image of B. Hence, $\omega_0 = 0, \omega_1 = 0$. Let $c \in \mathcal{F}$ be the constrained face, and θ_0 the constrained value in Equation (2). Now, set $\theta = \omega_2 - (\omega_2(c) + \theta_0)\mathbb{1}$, where $\mathbb{1}$ is a constant vector with all ones. Then, $\theta(c) = \theta_0$ and $d_1^T \theta = d_1^T \omega_2 = \omega$, because $d_1^T \mathbb{1} = 0$. Furthermore, θ is defined uniquely, since ω_2 is unique up to a constant shift. Thus (p, θ) is the unique solution of Equations (14), (15) that is also feasible for (2).

Theorem 3.1.

(i) Let (θ, p) be a feasible solution of (2). Then, for any integral basis of non-contractible dual cycles H, there exists a feasible solution (x, α, β) of (5) such that $E_T(x) = E_M(\theta, p)$.

(ii) Let (x, α, β) be a feasible solution of (5). Then, for any dual spanning tree $\mathcal{T}$, there exists a feasible solution (θ, p) of (2) such that $E_M(\theta, p) = E_T(x)$.

(iii) Let (x, α, β) and (θ, p) be corresponding solutions as in (i,ii), and let θ_T be the integrated values of x. Then $\theta_T = \theta \mod \pi/2$.

Proof.

(i) Let (θ, p) be a feasible solution of (2), and let (x, α, β) be the feasible solution of Equation (5) guaranteed by Lemma 8.3. Then $-x = d_1^T \theta + r + \frac{\pi}{2} p$, due to Equation (15). Thus $E_M(\theta, p) = \| d_1^T \theta + r + \frac{\pi}{2} p \|_2^2 = \| -x \|_2^2 = E_T(x)$.

(ii) Let (x, α, β) be a feasible solution of (5), and let (p, θ) be the feasible solution of Equation (2) guaranteed by Lemma 8.4. Then, Equation (14) again leads to $E_M(\theta, p) = E_T(x)$.

(iii) Let (x, α, β) and (θ, p) be corresponding feasible solutions given by Equations (14), (15). Then, θ_T, the integrated angle values, are computed by $d_1^T \theta_T = -x - r$. Furthermore, from Equation (15) we have that $d_1^T \theta = -x - r - \frac{\pi}{2} p$. Thus, $d_1^T (\theta_T - \theta) = \frac{\pi}{2} p = 0 \mod \frac{\pi}{2}$. Since on the constrained face $c \in \mathcal{F}$ we have that $\theta_T(c) = \theta_c = \theta_0$, we get that $\theta_T - \theta = 0 \mod \frac{\pi}{2}$.

Lemma 8.6.

(i) If (x, α, β) is an optimal solution to Equation (5) then $x = d_0 a(\alpha) + Bb(\alpha, \beta)$.

(ii) If $x = d_0 a(\alpha) + Bb(\alpha, \beta)$ then $E_T(x) = E_I(\alpha, \beta)$.

(iii) If (x, α, β) is an optimal solution to Equation (5) then (α, β) is a feasible solution to Equation (11).

(iv) If (α, β) is a feasible solution to Equation (11), then for $x = d_0 a(\alpha) + Bb(\alpha, \beta)$, we have that (x, α, β) is a feasible solution to Equation (5).

Proof.

(i) Let (x, α, β) be an optimal solution of Equation (5). Then there exists a unique decomposition (up to two constants) $x = d_0 a + Bb + d_1^T c$. Since x is optimal, the matrices d_0, B, d_1 are mutually orthogonal and c is not constrained, then $c = 0$. Due to the constraints in (5) we have that $d_0^T d_0 a = \frac{\pi}{2}\alpha - \alpha_g$, thus $a = a(\alpha) + c\mathbb{1}$ for some constant $c \in \mathbb{R}$. Hence, $d_0 a = d_0 a(\alpha)$, since $\mathbb{1}$ is in the kernel of d_0. In addition, also due to the constraints in (5), we have $H^T d_0 a + H^T B b = \frac{\pi}{2}\beta - \beta_g(\Gamma)$, thus $b = b(\alpha, \beta)$.

(ii) This is trivial, since d_0, B are orthogonal.

(iii) This is trivial, since the constraints of (11) are a subset of the constraints of (5).

(iv) Let (α, β) be an optimal solution of Equation (11). Then $\alpha^T \mathbb{1} = 4\chi$, and $(\frac{\pi}{2}\alpha - \alpha_g)^T \mathbb{1} = 0$. Therefore, we have: $d_0^T d_0 a(\alpha) = LL^\dagger(\frac{\pi}{2}\alpha - \alpha_g) = \frac{\pi}{2}\alpha - \alpha_g$, since $LL^\dagger m = m$ for any m orthogonal to the kernel of L. Furthermore, $H^T d_0 a(\alpha) + H^T B b(\alpha, \beta) = H^T d_0 a(\alpha) + \frac{\pi}{2}\beta - \beta_g(\Gamma) - H^T d_0 a(\alpha) = \frac{\pi}{2}\beta - \beta_g(\Gamma)$. Therefore, if we set $x = d_0 a(\alpha) + Bb(\alpha, \beta)$, then (x, α, β) fulfills the constraints in Equation (5).

Theorem 3.3. *(x, α, β) is an optimal solution to Equation (5) if and only if $x = d_0 a(\alpha) + Bb(\alpha, \beta)$ and (α, β) is an optimal solution to Equation (11).*

Proof.

⇒ Let (x, α, β) be an optimal solution of Equation (5). Then $x = d_0 a(\alpha) + Bb(\alpha, \beta)$ by Lemma 8.6 (i). Assume that (α, β) is not optimal for Equation (11). Then, there exists an optimal solution $(\tilde{\alpha}, \tilde{\beta})$ such that $E_I(\tilde{\alpha}, \tilde{\beta}) < E_I(\alpha, \beta)$. But then, take $\tilde{x} = d_0 a(\tilde{\alpha}) + Bb(\tilde{\alpha}, \tilde{\beta})$. According to Lemma 8.6 we have that $\tilde{x}$ is feasible, and $E_T(\tilde{x}) = E_I(\tilde{\alpha}, \tilde{\beta}) < E_I(\alpha, \beta) = E_T(x)$, contradicting the optimality of (x, α, β).

⇐ Let (α, β) be an optimal solution to Equation (11), and set $x = d_0 a(\alpha) + Bb(\alpha, \beta)$, then according to Lemma 8.6, (x, α, β) is feasible for Equation (5). Assume that it is not optimal, then there exists an optimal solution $(\tilde{x}, \tilde{\alpha}, \tilde{\beta})$ such that $E_T(\tilde{x}, \tilde{\alpha}, \tilde{\beta}) < E_T(x, \alpha, \beta)$. But since $(\tilde{x}, \tilde{\alpha}, \tilde{\beta})$ is optimal, we have that $\tilde{x} = d_0 a(\tilde{\alpha}) + Bb(\tilde{\alpha}, \tilde{\beta})$, and therefore $E_T(\tilde{x}, \tilde{\alpha}, \tilde{\beta}) = E_I(\tilde{\alpha}, \tilde{\beta})$. Hence, we have $E_I(\tilde{\alpha}, \tilde{\beta}) = E_T(\tilde{x}, \tilde{\alpha}, \tilde{\beta}) < E_T(x, \alpha, \beta) = E_I(\alpha, \beta)$, contradicting the optimality of (α, β).